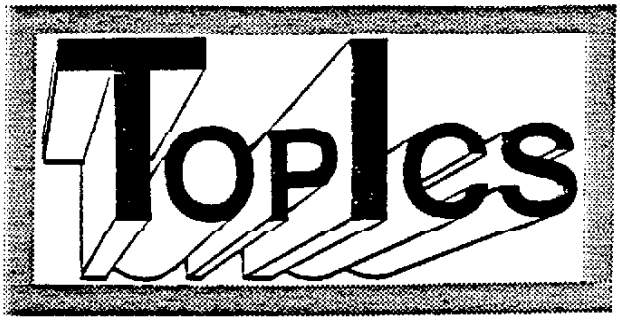
99'er FEST-WEST '88
SAN DIEGO
FEBRUARY
18 - 19
PLAN AHEAD!
BE THERE



LA 99ers COMPUTER GROUP Newsletter

VOL. 7 NO. 9 LOS ANGELES CA SEP 1988

TERRIES CORNER

WHITE HATS
JZ aka Jonathan Zittrain
Courage, bravery, no yellow stripe running down his back.
TIGERCUB aka Jim Peterson
This incredible man is still at great personal expense sending disks full of tips to User Groups recognizing his value to the community.

I'd give out a Black Hat too, if, I could figure out how to fit it on an empty head.

Disa N Data
When it came around to election time (beginning of this year), we were so underwhelmed with volunteers we decided to stay intact until mid year. Hoping of course by that time some would realize how very much they wanted to be an officer of this club. Well, we are sort of back to where we were couple of years ago. Tom Freeman has chosen to take a break from his Presidency role for a few reasons, personal, work load and a desire to devote more time to programming. Tom alone was double column formatting this newsletter. It is a labor intensive job, and as you can see I have not learned to follow in his footsteps, so what you see is what you get. Actually Tom got me out of a bind tonite and I thank him for that. "that's what friends are for". So by our by-laws vice becomes president, and here I am again. I do not need to have this honor and am very willing to hand it over to a willing volunteer. Anybody?

Tom, thanks to Continental, Jim Lohmeyer and myself, thanks to Peter Gleed, and Peter flew to New York to see George. We found him back in the hospital after a set-back due to extreme dehydration caused by nausea. (take the damn medicine George!) We enjoyed the caring hospitality of George's brother Richard and sister Mim. Even gave us a car to use. George has a great positive attitude and we are all for it. He really enjoyed our visit, but was very happy to se his daughter Susan. As Steve Mehr mentions, we mentioned at our last meeting that we wanted to give George a present of a visit by his daughter. Persons present were very generous. We welcome any donations by those not present as we were a bit short of our costs (about 125.ØØ). Thanks.

Craig Miller Revisited

The following is an excerpt from the book Night Mission, published and copyright 1985 by MG, and is reproduced herein by permission.

The Power of And, as explained by Craig Miller, has been, I feel, the most consice text on the subject. For that reason, I decided to contact Mr. Miller for permission to reproduce it in part in this newsletter. The Night Mission package, including a book (90 pages of priceless information) and diskette which contains the game Night Mission, is available from the L.A. 99er's Computer Club Marketplace for only $18.50. The book: is required reading, and includes programming information and helpful tips not found anywhere else. The game: is sure to challenge you as it performs as if it was written in Assembly. Thank you Craig Miller, for continuing to be a part of the TI community. Steve Mehr.

THE POWER OF AND

One of the most powerful and versatile functions in Extended Basic and many other languages is the logical expression of AND. Unfortunately it has also been one of the least explained functions. In the TI Extended Basic manual they devote all of two and a half confusing pages to the logical expressions of AND, OR, XOR and NOT. This is also true for most of the other computer's basic manuals.

So a few years back we decided to find out all we could on this mysterious unexplained function. After much research and a number of books on Boolean Algebra and Boolean Logic we ended up more confused than when we began since most of the material did not deal with the subject on a computer level.

Well it was now time to tackle this subject on a purely experimental basis with Extended Basic. After many weeks of testing, trying, discovering and failing the answers finally became understandable and explainable in plain english. So now we would like to share our understanding of the use of AND on direct numbers with you.

You are all probably familiar with the use of AND with two relational expressions such as: IF A=1 AND B=2 THEN..... so we will not discuss this use. We will instead discuss the many uses of AND on direct numbers such as: C=C+1 AND 7 or C=C AND 32767 or IF C AND 1 THEN..... Used in this form AND works fast and usually reduces the amount of code, or bytes, in your program.

To start with lets look at a few of the possible uses of AND in your programs:

1. Very good for auto-reset counters that never increment beyond a certain value.

2. It can be used to easily determine if a number is Odd or Even.

3. Excellent for small pseudo random numbers when sprites are in motion and your program uses CALL POSITION.

4. Very good for conserving on the total number of variables in your program when you are using the variables as flags or condition indicators.

To be continued...

AARRRRRRRGHHHH!

by Steve Mehr, UG Member

There is so much going on in our community that I've just got to scream (see title). Our guest speaker this month was Peter Gleed from the Melbourne User's Group in Australia. What a treat it was to get a first hand account of our brother and sister TI'ers from "down under." Does it mean that we're from "up over?" Peter didn't come empty handed as he presented Fred Moore with the latest version (as of this writing) of Funnelweb, version 4.12, and a program written by his User's group, International Code Flags, which offers information and graphic displays of nautical flags. Very interesting! Thanks Peter, for your contributions to our meeting and to our library.

Next up was myself showing what new things are happening in the field of sound processing for the TI. No clues yet but... you'll see things you've never seen the TI do at the next meeting. GUARANTEED!

Ray Kazmer, no newcomer to the TI community, was there to enlighten us on the story of Sister Pat Taylor. (See the July '88 issue of "KAZMERpendium", Feedback section, for more information, and the August '88 issue, Feedback section, for a word from Sister Pat). Way to go Ray! Ray sends a big THANKS to all who contributed $$$ to help Sister Pat.

Making the User's Group circut lately is Colin Mahoney, Treasurer of the San Fernando Valley 99er's User's Group. His demonstration of TI-Base was very enlightening and showed the power of this latest data-base contender. Thank you Colin, your demonstrations are always interesting and well prepared.

Tom Freeman and Jim Lohmeyer gave us a look at Myarc's new Hard Floppy Disk Controller card. When I win the lottery <grin> I'm sure to purchase it and the four hard disks it's capable of running. Seeing the possibilities now available for our trusty TI helps to increase its life expectancy many fold. For those of you new in our community, your heads needn't spin over all the "high tech" talk you read between these pages. (Certainly not in my column!) Just remember that when you're ready to expand your computer system, there will be many, many roads to choose from and not only one way to go. Just one more reason to stay with the TI.

A special thanks to all who contributed $$$ to help George Steffen's daughter Susan Lehrer make the trip up to N.Y. from Lake Tahoe to see him. George, all our prayers are being sent with Susan! It's nice to see just how generous our membership is when called upon to help our fellow members. Thanks to all!

It was nice to see Mary Phillips from the Ozark 99er's User's Group again. Had there not been so many presentations, we would have had you speak for an hour or so! (You did come prepared, didn't you?)

Since I am skipping around so much I will mention what else you may see at the next meeting. Steven Doran will be present to demonstrate a space shuttle simulation program he has written. Should be interesting. Be there!

If anyone has something that they think might be of interest to the membership, either to demo yourself or as an idea for a future meeting, PLEASE let me know. Although it's not Star Search, you may still get your chance in the LA/4A limelight. Thank you.

BEGINNING FORTH #4 By Earl Raguse

MORE EDITOR CHANGES

Last time in BF#3, I gave you an improved Editor. This month we will add a new word APPEND to Screen #41, line 7 and extend line 10 of Screen #38 to call for it. APPEND is a remarkable word for its short length.

What APPEND does, is to permit you to append a line or part thereof, deleted with FCTN 3 or FCTN 7, to the right of the cursor on any line, on any screen! All you need to do is to get to the desired screen, if not the one you are on, place the cursor where you want the moved text to start and press CTRL 9 and its done. You may execute all sorts of words or key strokes in between, as long as you do not disturb PAD. Now that's real editing power. All that because I hate to type, well not really I suppose, or I wouldn't be doing this article, but I do hate to type anything I can make the computer do.

How did this all come about? Well one day I discussed the possibility of such a word with my friend and Forth mentor Lutz Winkler of San Diego, and he said, "That doesn't sound too tough, let me look at it". Almost by return mail, I had it, not only that but a new version of the autorepeat cursor and Inverse Video for the character under the cursor. After I tried that, I decided I didn't like the cursor movement and the Inverse Video, although many people might think it great. I studied his technique and decided I could take just the part I originally wanted and add it to Pete Korner's version of Editor, Screen #41 from BF#3. Fortunately it worked just fine, and thats what I have attached hereto. I had to move existing words around on Screen #41 to make room for APPEND which is on line 7. Also I had to add to line 10 of Screen #38 to call for APPEND. The new line 10 is;

```
10 7F OF -TAB ENDOF 1F OF APPEND ENDOF
```

After you make the above changes, if you BSAVEd your dictionary last time as I suggested, you will have to FORGET EDITOR1, and reload -EDITOR and whatever else you had loaded, then BSAVE it again.

Last time I had my BLUNDERBUSS out again and told you to FORGET BOX. That's the word to forget alright, but its not in Forth's vocabulary, its in EDITOR1's vocabulary, hence you must FORGET EDITOR1 instead. If I make another dumb mistake in the future (I probably will), and it gives you trouble, don't despair, call me at 714/847-5875, or corner me at the meetings.

DEFINING WORDS AND LOOPS

Now that we gotten over the interruption to our plan, lets get on with it. This time we will learn to define new Forth words. We will also use some definite loop control words like DO and LOOP. I had planned to do more with loops, but the unplanned Editor changes take up too much space, so next time I will continue with +LOOP and LEAVE and some indefinite loop control words BEGIN UNTIL WHILE REPEAT.

DEFINING NEW FORTH WORDS

Defining new Forth words or redefining old Forth words is so easy you can get carried away and overdo it. When I first was learning Forth, I was tempted to define a lot of new words which just combined existing

Forth words to do something which I thought at the time was very useful. Some did stand the test of time, but most did not. In the beginning one does not have much foresight, but its fun and almost harmless as long as you don't try to sell the world on your wonderful inventions that sometimes backfire.

One of the major powers of Forth is that if you don't like the way it works, or you don't like the words as written, you can change them to suit yourself. Not many other languages can do that. BASIC allows you to define Functions, but they are very slow, and you can't call them from other programs unless you repeat them. However, once you define a word in Forth, it is on a par with other resident words, it simply becomes part of the language. You must save it to disk of course if you want it to be available after you turn off the computer

The : (colon) is a Forth defining word, it signals Forth that the word following is to be compiled into the Dictionary along with the necessary addresses to access and execute the words following. The words following the newly defined word must have been previously defined, the definition's end is signaled by the ; (semicolon). Pretty easy, lets do one, but first lets look at a few existing Forth words.

The word CLS clears the CRT display. The word EMIT takes a number from the stack and prints it on the CRT, if it is the ASCII code of a printable character, else it prints a white blob. The word GOTOXY takes two numbers off the stack and uses them for X and Y coordinates on the CRT. The CRT has X=40 maximum characters (columns) horizontally and Y=24 maximum characters (rows) vertically.

We now define a new word STAR by entering the following, spaces are important, in the so called immediate mode, (ie Forth is booted and you have a cursor).

```
: STAR 20 12 GOTOXY CLS 42 EMIT CR ;
```

What do you suppose STAR does? Enter STAR and try it. It should go to the center of the screen (CRT), clear it, and print an asterisk (ASCII 42). Now wasn't that easy? You are right though, this is not one of the words which we are going to save for posterity.

When we do wish to save newly defined words, we should pick an empty screen, then CLEAR it if necessary. Please read Chapter 3 of the TIFM again. Then type your definition and when satisfied, enter FCTN 9 (Back) to exit EDITOR then FLUSH to save it to disk. Then to execute the word, enter the Scr# and LOAD, when the cursor returns, enter your word and hope for the best.

If it doesn't work as expected, EDIT the screen again. When you LOAD it again, Forth will alert you that your word "IS NOT UNIQUE" unless you remembered to FORGET it before LOADing. No problem, it just uses up memory, and I will show you a easy way to avoid this later. The last version of your word is what will execute. There is an excepton, suppose STAR is called by DOIT, another previosly compiled word, in that case, DOIT will execute the version of STAR in force at the time DOIT was compiled. In the immediate mode, if your word does not work right, you cannot edit it, all you can do is FORGET "word" and type the

whole thing again.

STAR may be a very good word for some application, but its too limited, it does just one thing, we could make it more flexible my leaving out the numbers and entering them on the stack before executing STAR.

Suppose we defined STAR as follows;

```
: STAR ( ch r c --)
       GOTOXY CLS EMIT CR ;
```

The (ch r c --) specifies that CHaracter code, Row (y) and Column (x) must be on the stack when STAR is executed, and the -- signifies that these numbers will be gone after execution.

Notice that when numbers are part of a word, they are entered on the stack in left to right order, ie last on top. When you put numbers on the stack for use by a word, you must put them in correct order, first used last entered, ie a LIFO stack. Think about it, and try it out.

STAR is now a more flexible word which could print any ASCII character anywhere on a cleared CRT. The trouble with this approach is that it seems kind of dumb to call a word STAR if we are going to print Q's, K's, or even spaces with it; and what if we didn't want to clear the screen every time, after all that would limit us to one character on the CRT at a time. Also are we sure we want a Carriage Return after each print? It begins to look like we didn't plan too well. It sure looks as if we should not combine these particular words into one new word just to be more efficient, unless, of course, our major purpose in life is to write single Stars anywhere on the CRT.

One of the commonly used Forth rewrites is to change GOTOXY to AT as follows.

```
: AT GOTOXY ;
```

You will find that AT is very frequently used, so a lot of keystrokes and screen space can be saved by it. CLS, CR and EMIT are usually used separately as needed. CLS and CR are frequently combined into words that print prompting messages. AT is what I call a Useful Forth Word, and I have attached several of my favorites. I will refer to them as UFW's.

One of the more cumbersome Forth words is EMPTY-BUFFERS, which clears all of the screen editing buffers. If a screen is already in the buffer, and you have made changes on it, but wish hadn't and since you have not yet FLUSHed the changes to disk, you might think you could simply enter the screen number and EDIT, to reload the old version. Well you can, but it won't do you any good because Forth will look in the editing buffers and find it there and just print that to the CRT instead of going back to the disk. If you enter EMPTY-BUFFERS first however, Forth clears all five screen editing buffers and loads in the specified screen. This long but frequently used word is often abbreviated to one of the following, ZAP, E/B, EB, MTB, etc. Take your pick. I like ZAP, which can be defined as follows;

```
: ZAP EMPTY-BUFFERS CLS 12 12
      AT ." BUFFERS ARE EMPTY" ;
```

We have introduced an important Forth word ." (dot quote) which prints to the CRT everything to its right until another set of quotes " is found. Notice that AT ." is much like XBASIC DISPLAY AT.

I have included here, copies of my favorite Forth Abbreviations as Screen #2 and my favorite UFW's as Screens #88 and #89. I will not discuss all of them this time, but I will talk more about them as we need them. If you can figure out what they do, use them, some are easy, some not. As you might expect, I make big use of E; , F; , L; , FG , SC , MV , VL and my lastest UFWs MVF and LS.

I suggest you type them all and incorporate them into your dictionary. If you LOAD these screens last after all other necessary stuff, you can dump them all with FG DEC if memory runs low. FREE lets you check memory remaining. Once loaded, I predict you will have to be very desperate before you do that.

Handy as the Forth Abbreviations are, they are intended for your use only for privately conversing with Forth. If you use these things on screens which others must interpret, you will be degrading the transportability of Forth. They are great time and finger savers, and take up very little memory. I hate to type anything I can make the computer do for me.

LOOPS

Forth DO loops do the same job as FOR NEXT loops in BASIC, and are of the form;

```
L I DO . . . . . LOOP
```

Loops require a limit L and a starting value for the index I in front of the word DO, then come the words to be repeatedly executed (none are required, as is the case for WAIT below), LOOP increments the index I, tests it against the limit L and if I less than L, action returns to just after DO, else it returns control to the word following LOOP.

Because of the way LOOP works, the number of times through the loop is L-I-1. Thus you must ajust L to get the desired number of passes through the loop. The value of L must be larger than I, unless we use +LOOP, to be discused next time. +LOOP can increment I by any number just like STEP in BASIC FOR NEXT loops.

The word WAIT uses a very elementary loop, it serves as a delay loop in the same way you would use a FOR NEXT loop in BASIC. The limit is not a part of the definition so you can change the delay by putting a number on the stack before executing WAIT. It takes about 1/7200 of a second for Forth to execute an empty loop. Thus if a number, say 2, were on the stack, it would be mulitplied by 7200, the product 14400 would become the DO limit L. We would thus get a 2 second delay. The word MS works the same way except the delay in miliseconds is put on the stack. The word CLRSCR is also a very useful looping word. It accepts a starting screen s1 the ending screen s2, adds 1 to it, SWAPs them and they become the DO limits. The loop CLEARs the Ith screen, then I . prints the screen number.

```
SCR #41
 0 ( EDITOR REPEAT KEY ROUTINE Pete Korner 12 3 84 )
 1   BASE->R DECIMAL 0 VARIABLE MY
 2 : BLINK CURPOS @@ DUP VSBR MY C!
 3   3 0 DO DUP 30 SWAP VSBW LOOP MY C@@ SWAP VSBW ;
 4   4 CONSTANT WW ( REPEAT SPEED) 30 CONSTANT XX ( DELAY)
 5   0 VARIABLE YY XX VARIABLE ZZ 0 VARIABLE OK
 6
 7 : APPEND PAD PTR C/L R/C DROP - CMOVE RELINE ;  newline
 8
 9 : RKEY BEGIN ?KEY -DUP BLINK BLINK
10   IF YY @@ 1 YY +! IF ZZ @@ YY @@ < IF WW ZZ ! 1 YY !
11   1 ELSE OK @@ OVER = IF DROP 0
12   ELSE 1 DUP YY ! ENDIF ENDIF ELSE 1 ENDIF
13   ELSE XX ZZ ! 0 YY ! 0 ENDIF UNTIL DUP OK ! ;
14 R->BASE
15

SCR #2
 0 ( ABBREVIATIONS & UFW'S ETC  EGR 7 11 87 )
 1 : DEC  DECIMAL ;     : DU DUMP ;        : 2/ 2 / ;
 2 : FC FORTH-COPY ;    : F;   FLUSH ;     : 2* 2 * ;
 3 : UD UPDATE ;        : FG   FORGET ;    : W; WHERE ;
 4 : CO COLD  ;         : SM   SMUDGE ;    : E; EDIT ;
 5 : VL VLIST ;         : TX   TEXT ;      : L; LOAD ;
 6 : AT GOTOXY ;        : SC SCOPY ;       : C; CLEAR ;
 7 : HOME CLS 0 0 AT ;  : MV SMOVE ;       : A; ABORT ;
 8 : MVF ( move a set of screens forward to overlapping area )
 9       ( start# end# move#) ROT ROT DUP ROT - 1+ 0 DO CR OVER
10       OVER I + - OVER I - OVER OVER . ." --> " . SWAP SCOPY
11       LOOP DROP DROP ;
12 : PL ( page length) SWCH 27 EMIT 67 EMIT EMIT UNSWCH ;
13 : SET ( printer setup) SWCH 27 EMIT 33 EMIT 9 EMIT
14       27 EMIT 65 EMIT 9 EMIT UNSWCH 88 PL ;
15 : LS UPDATE FLUSH SCR @@ LOAD ;

SCR #88
 0 ( EGR's USEFULL FORTH WORDS #1 REV 4  7 11 87 ) 2 CLOAD LS
 1 : BS  ' TASK 20 BSAVE . ;
 2 : ZAP EMPTY-BUFFERS CLS 10 12 GOTOXY ." BUFFERS MT" ;
 3 : WAIT  ( SEC---) 7200 * 0 DO LOOP ;
 4 : MS ( milisec wait) 5 * 0 DO LOOP ;
 5 : LF  ( lines to feed) SWCH 0 DO CR LOOP UNSWCH ;
 6 : FF  ( formfeed) SWCH 12 EMIT UNSWCH ;
 7 : .VL SWCH VLIST UNSWCH ; ( dictionary to printer)
 8 : P ( SCR#---) SET SWCH LIST UNSWCH ; ( print screen#)
 9 : .HD  ( U ) SWCH 25 SPACES ." DISK: " . CR UNSWCH ;
10 : CLRSCR ( s1 s2 ---) 1 + SWAP DO I I . CLEAR LOOP ;
11 : PRTSCR ( s1 s2 ---) 1 + SWAP DO I I . P LOOP FF ;
12 : FREE SP@@ HERE - . ." BYTES LEFT" ;
13 : PS SET PRTSCR ;
14 :  ( skip to end) IN @@ 64 / 1+ 64 * IN ! ; IMMEDIATE
15   -->


SCR #89
 0 ( EGR's USEFUL FORTH WORDS #2 REV 4  7 11 87 )
 1 : INDXHD  SWCH 24 SPACES ." INDEX FOR DISK " . CR UNSWCH ;
 2 : CAT ( u1,u2,Dsk#---) INDXHD 2 LF SWCH INDEX UNSWCH ;
 3 : XPECT ( u--) S0 @@ SWAP EXPECT 0 IN ! ;
 4 : GSTR$ ( u--) XPECT 1 PAD 72 BLANKS PAD HERE - 1 -
 5     DUP ALLOT MINUS SWAP WORD ALLOT ;
 6 : CSTR$  ( adr1 adr2 u) OVER OVER + ROT DO DROP 1+ DUP 1-
 7   C@@ I C@@ - DUP IF DUP ABS / LEAVE ENDIF LOOP SWAP DROP 0= ;
 8 : GNUMB QUERY INTERPRET ;
 9 : PP  PAD 72 -TRAILING TYPE ; ( print the PAD )
10 : .ASK CLS 2 12 AT ." Again? Press SPACE else ANY other" ;
11 : PAK 12 16 AT ." PRESS ANY KEY " KEY DROP ;
12 : I> COMPILE R> ; IMMEDIATE  : >I ( a) COMPILE >R ; IMMEDIATE
13 : UNDO I> R> R> DROP DROP >I ;
14
15
```

Did you know that...?

by Chick De Marti

SEPT 1988

IS THERE NOTHING SACRED?

Up until now, APPLE has been free of clones, but now (for those of you who can't make up your mind weather your second computer (TI being #1) should be an APPLE or an IBM), the Cardata Co. of Compton, Ca. is producing the WPC Bridge, a new microcomputer that emulates a 128K Apple II and a 512K IBM PC-AT clone. WPC (WIZARD PC) and should retail for about $1695, will include a built-in 12 inch tilt screen monitor, two 5 1/4 drives and a plug-in keyboard. It also includes 3 PC-style expansion slots, parallel and RS-232 serial ports and an APPLE 9-pin game port.

Now you can have your Apple,cake, and eat it, too!

~~~~~~~~~~

MORE CALL LOADS

For TRUE random numbers, install this line into the load program:

```
CALL PEEK(-31880,A,B):: CALL
INIT :: CALL LOAD(-31808,A,B)
```

~~~~~~~~~~

TRY:

```
100 ON BREAK 4000
4000 CALL INIT :: CALL LOAD(-
31952,255,231,255,231)
```

(This erases the program itself from memory without erasing the screen... also without disturbing assembly routines in lower memory.)

~~~~~~~~~~

I talk about using "meaningful names for variables in my EZ-BASIC tutorial but this can lead to surprises!

```
100 FOR MOVE=1 TO 10
110 PRINT MOVE,
120 NEXT MOVE
```

Works fine in Basic and XBasic, but try it in Super Extended Basic and you get "ILLEGAL COMMAND IN PROGRAM" because apparently the word "MOVE" is reserved!

~~~~~~~~~~

This next item came under the by-line CALL QUESTION (but which U.G.?)

Q:
When writing programs, I sometimes find that I have to add a new variable in order to keep track of something which I had forgotten about. My problem is:
How can I tell if I have already used that variable name for something else (especially in some obscure part of the program where I might not notice it on a screen listing or printout)?

A:
Quick solution: temporarily DIMension the variable as an array. If you have used that variable name elsewhere as a simple variable, the computer will give an "IMPROPER NAME" error message when it is "pre-scanning" the program. So you'll know, if the program starts to run, you haven't used that name as a variable. Stop the program, take the DIM statement out, and use the variable. (By the way, reverse the procedure for an array. However, fewer arrays are generally used, and hence, are easier to keep track of.)

~~~~~~~~~~
~~~~~~~~~~

(Did You Know ... cont.)

WHAT'S A GAMER?
==========

Ever wonder about people who play the video games? What kind of people are "GAMERS"? These stats are from a survey by David Hallerman (see Family Computing, March '87).

77% are males
58% over 20
28% play on Commodore 64
27% Apple (these 3 have sold the
24% IBM bulk of the computers)
30% been computing 1 to 2 years
74% don't have any particular rules as to when or how to play.
84% play alone, single player games
81% play rather than watch TV
49% pay $20 to $35 per game
79% find out about games via magizine reviews and/or ads.
83% enjoy reading reviews of games (only 25% read "Interviews with designers" (of programs))
71% prefer Tricks and Hints

Arcade games are most popular.
Text only (strategy) games are the least popular.
ZORK is number 1
Flight Simulater is in 3rd place
Pac Man holds down number 10.

~~~~~~~~~~

VCR ... AS A BACKUP FILE?
==========

Did you know that a 2hr video tape can store 80 megabytes of computer data? So why not use it? The VIDEOTRAX tape backup system will dump your data to either VHS or BETA format, ...for only $595 for the controller card. (No! you cannot play the video tape...you have to load it back into your computer. And no! It's not available for the TI. Yet!

Here's one that...I don't know why, but I don't REALLY know how come. Can you figure out these statements

+(H=120) or *(M1=1)

Assuming: SC=SCORE and M1=COLOR (yellow)
Here's the line to decipher:

```
10 SC=SC+(H=120)*-50)+(H=112)
*-100((H=128)*(M1=1)0)
```

"H=120" means IF H=120, then the statement being true, H will =-1. Thus -1*-50=+50, or "H=128) sez if H does =128, and M=1, (is the item is yellow) then line 10 =a bonus of 250. Plain as mud! And I'm sure there is more to it than this. For instance:

+(H<120) or *(H)120)

Anyone know any good tutorials on this?

~~~~~~~~~~

```
10 ! *** BACKWARDS ***
20 ! A subroutine to print
30 ! something BACKWARDS
40 !
9000 INPUT WORD$
9010 BACKWARD$=""
9020 LENGTH=LEN(WORD$)
9030 FOR X=LENGTH TO 1 STEP
-1
9040 LENGTH$=SEG$(WORD$,X,1)
9050 BACKWARD$=BACKWARD$&LEN
GTH$
9060 NEXT X
9070 PRINT :BACKWARD$
9080 END ! ---tempory---
```

Well, I'm out of coffee. See you next month
Chick

E Z * BASICS

by Chick De Marti

EZ-BASIC (6)

Those of you who have been following our little data file tutorial, should be getting excited because we are now getting into the meat of the program. So far we have created a title screen (with some bells and whistles), a menu (using the "ON-GOTO" command, and created a "find the file" routine. We also tried a variety of DISPLAY AT commands. True, PRINT is easier to enter, but it has it's drawbacks:

1. It only PRINTs to the bottom of the screen.
2. which therefore creates scrolling.
3. And it starts on line 23 (not 24)! (There is a way to defeat this, but that's another story).
4. While you CAN PRINT something to the end of a line, you can't PRINT something to the middle of an existing line.

Try this XBasic routine:

```
1910 DISPLAY AT(24,2):"Press
     key to continue"
1920 FOR FLASH=1 TO 6
1930 DISPLAY AT(24,8)SIZE(-3
):"  "
1940 FOR DELAY=1 TO 50 :: NE
XT DELAY
1950 DISPLAY AT(24,2)SIZE(-3
):"KEY"
1960 FOR DELAY=1 TO 50 :: NE
XT DELAY
1970 NEXT FLASH
```

I think you'll agree DISPLAY AT if worth the effort.

This monthe we are going to add a SORT routine to our EZ-LIST program. If you haven't been following the tutorial, you can enter the following segment on it's own. It uses the BUBBLE SORT. I suggest you study the routine with me, for there are several items the may seem confusing.

For those of you who are follow the EZ-LIST program, I have numbered this segment so it doesn't interfere with your existing program. You can either type the SORT by itself, try it, then merge it into EZ-LIST, or you can LOAD EZ-LIST first,and add these lines directly to the program.

Some tidying up first:

* Renumber your existing DATA as lines 2000 to 2050
* Make the last DATA line 9000 DATA "ZZZ",,, (the commas are added so there won't be a visable delay, whenthe READ statement nears the end of data).
* I am using a slightly different version of the FLASH routine in EZ-SORT than the one listed above.

Now for some facts.

Because we want to keep track of one name, while comparing to to another, it is necessary to put our data into an array.

ARRAY

Arrays have been compared to mail boxes. The first item is put into box (address) #1. The next into box #2, etc. We have six different names. By putting each at a seperate address (box), we can keep track of them essily. For example, if Funke Allen is in box 3, anytime we need to do something with Mr Funke, we simply tell the computer to "do such and such with the information in 'box 3'"

In the TI System, anytime an array is going to be larger than 10 items, it is necessary to save space for all the items, before using them. If you don't, the moment you make reference to 'box' 11, you will receive a data error (usually in the line that is READing data). Although we only have 6 items of data, we can anticipate a phone list as being larger than 10 names (besides, it's a good habit to get into). The way to reserve space is to DIMENTION (or DIM) the array.

DIM

format DIM name1$,(size)<name2$,(size)

Most programmers DIM their arrays in the first few lines. Since we already had our EZ-LIST program started (and we are not using the SORT till later) I placed it at line 700 the first line of the sort routine (this also makes it easy for those who do not have EZ-LIST to use EZ-SORT on it's own...or they can easily add it to their own program).

```
700     CALL     CLEAR     ::  ! DIM
NAM$(50),DOB$(50),INFO$(50)
```

More information:

Our DATA files only contains a name and a date. See lines 450 and 460.

```
450 READ NAME$
460 IF NAME$="ZZZ" THEN 510 ELSE READ
DATE$
```

(EZ-BASIC continued)

The data "ZZZ" allows us to leave our READ loop prematurly.

At this point, note that we used the string variable NAME$ (for data name) and DATE$ (for birthdays). Because the same variable MAY NOT be used in a common variable and an array variable, I chose the variables NAM$ (close enough) and DOB$ (date of birth). Also, because I wanted to create one string to hold a name, and address, and anything else we might like to add, I created the variable INFO$. Lines 840 to 860 handles this task for us

The SORT routine is only9 lines long. Lines 880 to 960.

O.K. gang. Armed with this information, enter the EZ-SORT program. Try it, examine it, and when your satisfied (if you haven't already typed in into your original program), type **SAVE DSKn.EZ-SORT,MERGE**

Once it is saved (now in a MERGE format)enter **MERGE DSKn.EZ-LIST**

The two programs will be merged, all line numbers intertwined. Try the entire program, now with three choices, and if all works well, you can delete lines 722 to 744.

Enjoy yourself...see you next month...CHICK

(The EZ-SORT program follows)

```
700 CALL CLEAR :: DIM NAM$(50),DOB$(50),INFO$(50)
702 ! --Exam. of BUBBLE SORT--
704 ! --for EZ/LIST program--
706 !
708 RESTORE
722 !
724 ! ---print data as is--
726 !
728 PRINT "FILE AS IS": : :
730 FOR I=1 TO 50
732 READ NAM$(I),DOB$(I)
734 IF NAM$(I)="ZZZ" THEN 740
736 PRINT NAM$(I),DOB$(I)
738 NEXT I
740 TOP=I-1
742 PRINT :"SORTED FILE": : :
744 !
800 !
810 ! --sort routine--
820 !
830 ! CALL CLEAR :: RESTORE
840 FOR I=1 TO TOP
850 INFO$(I)=NAM$(I)&"/"&DOB$(I)
860 NEXT I
870 SORT=1
880 SWAP=0
890 FOR X=1 TO TOP-1
900 IF INFO$(X)<=INFO$(X+1)THEN 950
910 SWAP=1
920 B$=INFO$(X)
930 INFO$(X)=INFO$(X+1)
940 INFO$(X+1)=B$
950 NEXT X
960 IF SWAP=1 THEN 880
970 !
980 ! ---print Sorted file---
990 !
1010 FOR I=1 TO TOP
1020 P=POS(INFO$(I),"/",1)
1030 NAM$(I)=SEG$(INFO$(I),1,P-1)
1040 DOB$(I)=SEG$(INFO$(I),P+1,28)
1050 PRINT NAM$(I),DOB$(I)
1080 NEXT I :: PRINT
1090 GOSUB 1900 ! pause routine
1110 GOTO 240
1900 ! --pause routine--
1910 !
1920 FOR FLASH=1 TO 6
1930 DISPLAY AT(24,2):"Press ___ key to continue"
1940 FOR DELAY=1 TO 50 :: NEXT DELAY
1950 DISPLAY AT(24,2):"Press ANY key to continue"
1960 FOR DELAY=1 TO 50 :: NEXT DELAY
1970 NEXT FLASH
1980 CALL KEY(0,K,S):: IF S=0 THEN 1980
1990 RETURN
2000 DATA BENNETT TONY,10/04/32
2010 DATA TRENT HELEN,12/10/40
2020 DATA FUNKE ALLEN,01/27/29
2030 DATA BAKER LILLY,01/05/53
2040 DATA BUTCHER RED,10/20/60
2050 DATA BUTNER IVAN,06/03/48
2060 DATA "ZZZ",,,
```

Reviews

SUPERBASIC (c) 1987

This will have to be an abbreviated review because there are so many features available. If you haven't bought the "SUPER EXTENDED BASIC" cartridge yet, THIS is a must! Even if you have, this is a great addition to your utility library (altho SUPERBASIC will not work with your SUPER EXTENDED BASIC cartridge plugged in...both contain some of the same commands, therefore you get a syntax error when trying to run your new acquisition).

But what power, what varity, what convenience!

How would you like to type "DIR 2" (like the big boys) and get a catalog of programs in drive #2? Or better still, CTRL 2 does the same thing (CTRL 1 catalogs drive 1, CTRL 3 cats drive 3, etc up to 6 drives)!

How about reading a DV/80 file (with a pause available) by simply typing TYPE "filename" (similar to IBMs MORE DOS command)? But let's start with some great time savers. As we all know:

FCTN 1 - DELETES a character
FCTN 2 - INSERTS a character
FCTN 3 - ERASES a line
FCTN 4 - BREAKS a running program
FCTN 8 - REPEATS the last instruction.

But FCTNs 5,6,7,9, and zero are usualy reserved for the programmer's use, so why not use them? With SUPERBASIC,

FCTN 5 TABs backwards a half line
FCTN 6 TABs foward a half line
FCTN 9 CLEARS to the beginning of the line
FCTN 0 CLEARS to the end of the line
and FCTN 7 lists all the new commands available in SUPERBASIC. Such as you'll have amazing manuverability with:

* **DEL** m-n (or DEL m,n) deletes lines from m to n.

* **RENUM** start,finish,new (,increment)

* **JOIN** n will combine the next line to line n

* / (the slash)=FIND The format is:

/pattern Pattern being the item or items you want located.

EXAM. /GOTO 400 will find all occurrences of the command GOTO 400.

EXAM. /IF A$= will list all occurrences of this phrase.

You can also:

* **COPY** a DV/80 file to the same drive, or a different drive and change it's new at the same time.

* **COPY** a DV/80 file to your printer, to get a hard copy (it will honor all TI-Writer DOT commands).

* **APPEND** one DV/80 file to another.

* **RENAME** a file

* **LOCK** (or protect) a file, or:

* **UNLOCK** to unprotect a file.

* **QOFF** (QuitOFF) disables the FCTN-quit

* **QON** (QuitON) enables the FCTN-quit.

* **ENTER** "filename" will merge a DV/80 file into program memory, (allowing you to use TI-Writer's full screen editor to write, rewrite, or edit a program...great for Newsletter entries!

* **EDIT** "filename" allows you to edit a DV/80 file without leaving Extended Basic. When you finish editing, you can return it to the original file, create a new file, or just abort the project.

* **RECOVER** A seperate program is also included as an aid to recover any programs lost by a system crash, or the "NEW" command.

There are 6 pages of DOCS available on the disk. You will find the instructions complete and easy to understand, but a "KEY" (to be inserted into the joystick plug) is required. The files, minus the DOCS, only take up 135 sectors, and because the disk is not copy-protected, it would be a good idea to copy them to any disk you are using in XBasic as the program will auto-load and then remain in lower memory, leaving you RAM still available. And last but not least, with the CALL LINK("INSKEY",n,string) command you can create up to 32 softkey definitions, where n is the ASCII code for the key being redefined and string is the desired new key definition. Exams. of string might be:

```
DISPLAY AT(
CALL KEY(0,K,S):: IF S=0 THE
N
```

or even:

```
! ************************  and
!     by CHick De Marti
```

TELECOMMUNICATIONS

BBS'ing with Danny.

SysOp LA99ers TI-WORLD BBS.

One of the things that seems to give the users of BBS's a lot of trouble is the entering of messages on a BBS. Now before we go too far afield, you must understand that no 2 BBS's are exactly alike! The major thing you MUST DO, is to READ the PROMPTS. They are there to help you. On most boards they are, for the most part, the same. But even if they are not, the most important thing is to READ,use common sense and what ever you do DON'T PANIC!!! It will not bite or grab you and spank your behind if you make a mistake.

ENTERING and EDITING MESSAGES!

This is how it will look online. From this point to the end, is just what you will see on line without the notes that I have added to try to explain what is going on. If you read this and the July column, you should have no problems, signing on to and moving about the board.

From time to time you will see a symbol that I have placed in the text, to indicate the key press you are to make. They will look like this:

==> <==

Lets do it!!!

```
=======================================
 *>> TI-WORLD 99 BBS MAIN MENU <<*

   A)NSI graphics on/off
   B)ulletins
   C)hat with sysop
   E)nter new user info
   F)ile transfers
   G)oodbye
   H)elp
   I)nfo and news
   L)eave sysop message
==>M)essage base

   O)ther TI BBS's
   Q)uick exit
   U)ser log
   Y)our info
   <Enter> This menu
=======================================
       Choice: M <==
```

After pressing M you will next see this!

```
=======================================
        *TI-WORLD Message Base*

Base has messages 106 to 205

Highest you've read: 205

 D)elete E)nter R)ead S)earch Q)uit: E
```

==> To enter a mess. you press "E" at this point. Upper or lowercase will work.

```
Enter user#, OR 999 to search for user.
OR press enter for ALL

 To:   <== WILL ONLY TAKE # OR ENTER.
```

If you put in a # at this point, it will ask you if the mess is (PRIVATE)Y/N. If you press <ENTER> (for all) then it is not asked.

What is the subject?
(The following is an example of an entry).

```
DEMO FOR TOPICS <== 40 CHARS OR LESS

From: DANNY NELSON          * N O T E *
Los Angeles, CA. #1    When you've finished
                       entering your Message.
To: ALL                it will be shown again
                       to you. You are then
Sub: DEMO FOR TOPICS   asked, "is it OK?", to
==> (enter Msg, here)  which you will answer,
09/06/88               Yes,No or Quit.

OK? (Y/N/Q) Y <==
```

Now you will begin to enter your message and the prompt below will tell you all you will need to know. You DO NOT NEED THE TO PRESS ENTER WHILE ENTERING THE MESSAGE.

```
Enter Text: Ten 79 character lines
Press Enter twice to end.  <== NOTE
(Word-Wrap on).

This message is beino entered on the mes
sage base of the LA99ers. TI-WORLD BBS.
as a demo of how it works. Do this at yo
ur own pace.  There is no one to rush yo
u.  The only thino you must not do is to
sit there for 2 minutes without pressino
a key. If you make a misteke you will be
allowed to change it before you save the
message. To end this message I will pres
s enter on a blank line.  At know time,
durning the entering of this message hav
e I used the <ENTER> key.

 C)on S)ave L)ist E)dit A)bort:
```

(BBS'ing CONTINUED)

EDITING
=======

Looking at the message, you will see that I used "know" when it should have been "no". To edit this you must know what line number the word is on. At the command line, press:

C)on S)ave L)ist E)dit A)bort: L <==

1
This message is being entered on the mes
sage base of the LA99ers, TI-WORLD BBS,
2
as a demo of how it works. Do this at yo
ur own pace. There's no one to rush you.
3
The only thing you must not do is to sit
there for 2 min without pressing a key
4
If you make a mistake. You will be allow
ed to change it before you save the mess
age.
5
To end this message I will press enter
on a blank line. At know
6
time durning the entering of this messag
es have I used the <ENTER> key.

C)on S)ave L)ist E)dit A)bort:

We now know that the change needs to be made in line #5. So we now press:

C)on S)ave L)ist E)dit A)bort: E <==

You are now asked:

Which line #? 5 <==

After you have pressed 5 and ENTER, the old line scrolls up on the screen.

Let me NOTE here that you must RETYPE THE WHOLE LINE OVER AGAIN, and press <ENTER> at end of the line.

Old line 5
the message. To end this message I will
press enter on a blank line. At know
Enter new line
the message. To end this message I will
press <ENTER> on a blank line. At no

==> (You already noticed that on line 6, the word "durning" is spelled wrong. To correct this, (I also wanted to add some thing to this line, so we will have to repeat the precess. Enter at the command line:

C)on S)ave L)ist E)dit A)bort: E <==

Which line #? 6 <==
Old line 6
time durning the entering of this messag
es have I used the <ENTER> key.
Enter new line
time during the entering of this mess.
have I used <enter>. UNTIL NOW TO END.

Here I wanted to add more to the message, so I will now have to press "C" to continue.

C)on S)ave L)ist E)dit A)bort: C <==

Now I have pressed the C to add this line and the next one.
I will now wait until the time out comes on the screen, just so you can see it!

Last Line

When you get to line number 10 the above prompt will come up. A bell, Chime or beep will sound, telling you that you can enter no more in this message. You must take one of the actions below, EXCEPT CONTINUE. So press the "S" to save the message.

C)on S)ave L)ist E)dit A)bort: S <==

As I said in the last line of the message, I did not press a key for 1 1/2 mins. The board then gave me the warning below. This will happen in all areas of the board, except during file transfer, if you just set there without pressing a KEY. If you do not press a key within 30 seconds after the promot the program will automaticaly sign you off.

-- Press a Key to Avoid Timeout! --

D)elete E)nter R)ead S)earch Q)uit: Q <==

Pressing "Q" will now return you to the main Menu.

Speaking of Main Menus! I am going to do FCTN 9 and SF and return you to the rest of your News Letter. Until next time, KEEP ON BBS'ing: Danny

RELATIVE FILE SORTS in Extended Basic

By Bill Gaskill

When I first tackled the job of learning how to program in extended basic I did so because I wanted to be able to write applications specific enough to meet my individual needs. I purchased most any book available on basic or extended basic programming and set about the task of finding example programs that contained the various algorithms in them that I wanted in my applications.

When I first started to program I used the RAM based data file storage method because that was about all I could find in the example programs available to the novice 99er. But like most, I soon discovered that the 32K memory limitation of the 99 limited the size of my files to only a few hundred records at best. As this was not acceptable, I turned to relative or disk based data file storage because I could store many times the number of records in a file on disk than could be created in a RAM based environment. The problem came when I tried to find an example program that sorted a disk based data file. There just didn't seem to be any.

Being able to sort data is perhaps one of the most useful aspects of any program, but it is also one of the most difficult parts of programming to learn. Many articles about sorting data files have appeared over the years, but few if any of them dealt with how to sort a relative file. Most of the articles I have seen assume that RAM based data files are being used and most of the example programs assumed that the information in the file is captured in a one-dimensional array. This is nice for illustration, but hardly practical for the Extended Basic programmer who really needs a working sort routine for a large capacity data file. If you use a RAM based data file you are going to be limited to about 500 records in the file at best (with a one-dimensional array) and about 150 records in the worst case (with a multi-dimensional array). With a disk based data file you are limited only by the disk space available.

The program that accompanies this article shows one method of sorting a multi-dimensional, disk based data file. No doubt there are many other ways to get the job done, some perhaps more efficient. But this one works and it allows over 1000 records to be sorted if two disk drives are available. The two drive requirement exists because the sort program reads a portion of each original record into memory, sorts that data and then writes a new file based upon the position of the sorted data in the sort table that is created and the relative record number of each original record that is attached to the data in the sort table. A single drive can be used if the size of the new file plus the existing one does not exceed the available disk space. You just have to remember to use a different file name for the new file.

The idea for this program came from reading Ralph Molesworth's book on assembly language programming. In the chapter on sorting, Mr. Molesworth pointed out that one of the things that escapes many programmers when dealing with sorting algorithms is the fact that an entire string or record does not have to be used for a sort. Instead, only the portion that the record is to be sorted on should be used. The trick is to build a method of reference back to the original record after the sort so that the file can be read based upon the key item that was used for the sort.

For example, let's say that you were dealing with a file that had a field named DATE and you wanted to sort the file by date. The only data that needs to be sorted is DATE data. As long as you have a means of equating each individual date back to the record that each date came from, you can use only date data in the sort table. The impact of this is that you will be able to sort much larger files in the tiny 32K memory environment you are limited to.

In the sort program, which I have written to be as generic as possible, a FIXED file must be used, and it is assumed that the file has a REC ZERO that contains at least the file size information. Thus when the source file is OPENed in line 190 the FS value from REC Ø is read to determine the number of records in the file.

There are 8 possible fields in my sample file that may be sorted on. You can choose the field to sort on by specifying its number (1-8), and you can choose the length of the data in the selected field to sort by (SL). This means that you could sort by the first 5 characters or values in a field even if it had a total of 10 characters. Using fewer characters decreases the accuracy of the sort but it allows larger file capacities to be sorted.

In the example program I allow up to 1100 records to be sorted in a single file. Thus the DIM statement in line 110 lists 1100 in the A$ and A arrays used. The N$(8) array dimensions memory for the eight fields that exist in each record. The A$(1100) allocates RAM for the 1100 string data items that are read into memory for the sort and the A(1100) dimensions enough RAM to hold the relative record number of each record read in the file.

When SORT first opens it prompts for a file name to use and then a field on which to sort the file (SF). Line 170 asks for a "Depth of Sort:" (SL). The number that is entered here should not be higher than the length of the field you are going to sort the file on.

Line 190 opens the file specified and reads in the number of records that exist in the file based upon the value of FS.

Line 200 then reads each record from disk sequentially, taking into memory only that portion of each record that you specified;

```
A$(R)=SEG$(N$(SF),1,SL)
```

This is where the large file size sorts are made possible. Only a portion of the actual record is being read into memory and then sorted.

Line 200 also creates an array that keeps track of the relative record number for each record being read, to later use it in matching sorted data with the actual record that needs to be read from the source file and then written to the target file;

```
A(R)=R
```

When the contents of the file are in memory the sort begins. The entire sort algorithm is found in lines 220-310. The algorithm used is a modified Shell-Metzner sort that continually splits the file in half to make comparisons.

During the sort line 310 re-positions the data being sorted and also takes the relative record number of the data along with it. That is the key to the actual re-write for the target file that is created

later.

When the sorting is done (when U is equal to zero) the program prompts for a target file name. The DSK# and file name must be entered.

Line 360 then re-opens the source file while line 380 creates the target file. Line 370 is used only to segment the file name portion of I$ if you want to write it to REC 0 of the target file. If not, it may be eliminated.

Line 390 sets up the loop that will determine the number of records read from the source file and written to the target file. It also performs another critical task, that of telling the program which record to read next.

```
D=A(G)
```

Line 400 reads the appropriate record from the source file into memory.

Line 410 checks to see if the record just read is REC 0. If it is the record is not written to disk because it already exists from the PRINT #3 statement in line 380. Once D is not equal to zero the write of each record to the target file begins in line 420.

Note that REC D is read, but REC G is written. This allows the target file to be a sequential re-write of the source file with each new record written being the record read from the source file based upon the relative record number in the sort table. The sort table by the way is the A(1100) array.

When all records are written to the new file both it and the source file are closed in line 430.

While the SORT program is a total of 54 lines long it could be made even shorter. Much of what you see is cosmetic because I like to write clean, professional looking programs. The really important lines are those mentioned above. They are the coding that make things happen.

A practical application for this program might be sorting a data file created with John Taylor's popular Checkbook and Budget Management program.

To sort a CBM file you need only 7 elements in the array N$() and a maximum of 400 in A$() and A(). Those are limitations set by CBM.

Next you would have to add a second variable to the REC ZERO statements besides FS. CBM uses RC,SS in its REC ZERO variables. RC is the equivalent of FS in the SORT program. You needn't change FS to RC though. Just add SS so that the file reads properly.

To load the file you would enter DSK1., the word DATA and then the year and month of the file to be sorted. For example, if you wanted to sort the January 1988 checkbook file you would type in;

```
DSK1.DATA8801
```

as the file name to load. Then choose the desired field, such as 2 for DATE, enter a number no higher than 8 as the sort depth and away you go.

LA99 LIBRARY CORNER

TWO copies Of all program disks will be made available to the members at the regular meetings. If you plan to obtain any disks from the library at the meeting it is best to phone or write the LIBRARIAN in advance to be sure they will be on hand. I will put your name on them.

NEW POLICY All disks now cost $2.00 each. 5 for $10.00. All disks will be SSSD or DSSD (No fippies except mailing). Mailing charges first disk $0.50 each additional disk $0.25 USA. Outside USA double the cost.

If you have a **SSSD** drive be sure you get **both** A and B disk if the program is over 360 sectors (if available) other wise get the DSSD disk. It pays to have a DSSD drive.

0000A/B **LA99 DISKS LIBRARY CATALOG** SEPT 88 : $1.00 either SSSD(flippie) or DSSD.

NEW ADDS FOR SEPTEMBER LA99 LIBRARY

The Library Committee wish to give thanks to those who donated disk to our Library this month : Danny Nelson, Chick Demarti, Terrie Master, Earl Raguse, UGOC, Peter Gleed. H.U.G. Northcoast 99ers.

2207 **BEGINNING FORTH** By Earl Raguse 17161 Edwards ST. Huntington Beach, CA 92647 : Latest improvements for TI Forth (July 88). Music, ASCII table, Number Formating, Forth to DV/80, Decompilier, Arrays, Sorting, Files and many more. Menu driven and self explanatory. Loads E/A #3 DSK1.FORTH. SSSD(360)

2415 **FUNWRITER 4.12** Aug 1988 Fairware by Tony and Will McGovern 215 Grinsell St. Kotara, NSW 2288 Australia. Allows TI extended Basic or Myrac XBII to provide utility envionment for TI-WRITER, EDITOR ASSEMBELER AND DISK MANAGER with welcome improvements. Compatible with Geneve 9640. Need 32K disk drive + controller and printer. DSSD(701). 2415A(360) and 2415B(343) for SSSD drives.

2415C **FUNWRITER TUTOR** V1.2 By O.C.U.G. A supplement documentation for configuration instruction for Funnelweb 4.10, 4.11 and 4.12. And a TI Writer reference Guide chart. SSSD(213)

2665 **MASTER MENU** Shareware by William Gaskill. A loader to run basic, extended basic, image files and D/F 80 object files. written in A/E must have SXB(Triton) module to run. Automatic loads hold down space bar when disk starts to turn for SETUP file. 9 menus per chart 9 charts per disk. SSSD(127)

2666 **SORTASORT** By Newt Armstrong of O.C.U.G. Program teaches you how to write a SORT program by showing you how a sort programs works SSSD(24)

2667 **DSK/CAS** Fairware by R.A.GREEN This program is the Assem and Source codes + a number of other A/L Utilities . The main program will transfer programs from disk to cassett. better understand Assembly Language, sourse and object code to get full use out of this disk. Very useful. DSSD(616)

2668 **SYSTEM UTILITY** Fairware by Joe Nollan 908 E. 35th St. Tocoma, WA 98404 : A five part disk 1. Auto Loader, 2. System Utility, 3. Menu Maker, 4. Character Scrambler, 5.Title Maker. SSSD(346).

NEW ADDS SEPTEMBER LA99 LIBRARY

2669 **1000 WORDS** Fairware by Norman Rokkie 231 Woodridge Dr. Apt.B204 Wintersville, OH 43952 : Converts TI-ARTIST files_P to TI-WRITER D/V 80 files. This means that you can print a document (letter) using text formatter that has both text and graphic. Written in assembly and converts fast. SSSD(284).

2837 **DSKU/4.12** Fairware by John Birdwell 7052 Springhill Circle Eden Prarie, MN 55344 An update of DSKU. An excellent Disk Utilities program that has FILE UTILITIES, DISK MANAGER, DISK UTILITIES, SECTOR SYSTEM SET UP. Will work on Myarc Controller and Geneve 9640. DSSD(444) 2837A(235) and 2837B(211) for SSSD drives.

2841 **COLORTV** From O.C.U.G. The best color TV and monitor check program that I have seen. Dot Matrix Color Bars, Horz and Vert checks with instruction on screen for each test. X/B SSSD(34)

2842 **ASSEMBLER UTILITIES** By Garry Christensen Australia : Set of utilities for the Assembler. This file has most of the most used A/L Modules The sets are so that you can use them right off of the disk. DSSD(620). 2842A(327) and 2842B(291) for SSSD drives.

2843 **ASSEMBLY TUTORIAL** By Assembly made easy. Maybe the best tutorial of Assembly that has vome out. Very easy way to learn and put together your own prigram in A/L. DSSD(520). 2843A(213) and 2843B(289) for SSSD drives.

4541 **MISC #29** 18 short miscellanceous programs from H.U.G : DR.VERNE(psychoanalsis), CALENDAR(1981 to 2000), EASTER(date), BIORTHYTHMS(1 or 2 persons), DISK CATALOG, LOVE TI, CALENDAR(any year), LETTERS(large or giant), GRADE BOOK(student), SORT(shell or bubble), TEMPERATURE CONVERSION, SOUND EFFECT(bee fly chime auto etc.), FAMILY TREE(records keeping), CLOCK(12 or 24 hour). SSSD(351)

5043 **FLAGS** From Melbourne TI User Group : Learn the International Code Flags for communicating between ships. Great color and graphic of flags 1-9 A-Z. SSSD(246)

5044 **PLATO CATALOG** By Cy Leonard 48 Sugar Bear Dr. Safety Harbor, FL. Disk has catalog of all PLATO educational series. Will print list to printer or make a label of each program. Also has a program that will print a catalog of what is on a Plato disk. X/B uses 2 drives. DSSD(621). 5044A(293) and 5044 B(328) for SSSD drives.

5045 **BEGINNING BASIC** A program of 7 lessons written in DANISH. SSSD(264)

6035 **GENEALOGY** III version 2.3 Fairware by Walter Davies 17718 Orchard Lane Salinas, CA 93907 : Keep track of your ancestrals for 4 generation. Print a family group sheet, locate files with ease. This updated version has been greatly improved. (SSSD)282

6051 **DATEX** By Chick Demarti : A handy appointment calender. Displays the current month (see line 2) with options to view appointment, next month's list or choice of any other month. XB SSSD(32)

6052 **LOTTERY 99** By Dwayne Johephson : Make your own lottery by putting in your names and the program will pick a winner for you. SSSD(125)

NEW ADDS SEPTEMBER LA99 LIBRARY

9061 **GAMES #48** TI99-MONOPOLY VI.7 Fairware by Ross Mudie 47 Berowra Water Rd. Berowra N.S.W.Australia. updated version.Two or three players.Computer keeps score.SSSD(346)

9084 **TIC TAC TOE** By P.E.Schippnick Pomona, CA 91766 : 1 or 2 players, great graphic, computer keep track. Load to cassette or Mini Memory to RUN, instruction on disk. BASIC SSSD(74)

9085 **RAPID LOADER** V2.0 Fairware by Ray Kazmer 13225 Azores Ave. Sylmar, CA 91342 :A special assembly routine that will fast load 14 of Infocom adventure games. Must have the original Infocom game1 and game2 disks. SSSD(360)

9086 **NASTY** By Orphaned Instruments : A program that answers back to you in a nasty mode when your Extended Basic commands are inputed. Like "RUN" (answer why not walk). A silly program but here it is. SSSD(104)

LIBRARIAN FRED MOORE 7730 EMERSON AVE. LOS ANGELES,CA 90045 213-670-4293

THE 3rd WEST COAST LOGO CONFERENCE PRESENTS

HANDS ON TECHNOLOGY
CONFERENCE AND WORKSHOPS

Friday and Saturday, February 3-4, 1989

The Los Angeles Airport Hilton and Towers
Los Angeles, California

Co-sponsored by: Pepperdine University; Computer Using Educators, Inc.; Computer Using Educators, Los Angeles; International Council for Computers in Education and SIG Logo; and the Regional Educational Television Advisory Council (RETAC).

FALL-4-A-SHARE-FAIR

SWAP AND PICNIC for TI-99/4A and compatible computer enthusiasts.

October 9th, Sunday, 1988 - 12 noon to 5 pm. Free Admission. Cooperation and Sharing

Nine of the southern california TI Home Computer user groups are having a get-to-gather. We sincerely hope you will come.

Place: BACKS Community center in Placentia. It's beside a great park so bring a picnic snack. A soft-drinks table will be fully stocked.

Bring any programs and hardware you want to share or swap or sell.

We can pick-up some great bargins and discuss topics of mutual interest.

Commercial enterprizes participating: DataBioTics, Corcomp, Texcomp, Comprodine, T.A.P.E.Ltd, L.A.Group Marketplace, Data Depot (supplies), L & M Systems-MYARC.

300+ TI-99/4A Fairware program disks from 2 large local libraries will be available.

Consignment table: Bring your idle hardware and software to sell. Label it and leave it to be sold or sell it yourself. There will be plenty of used items from others for you to buy.

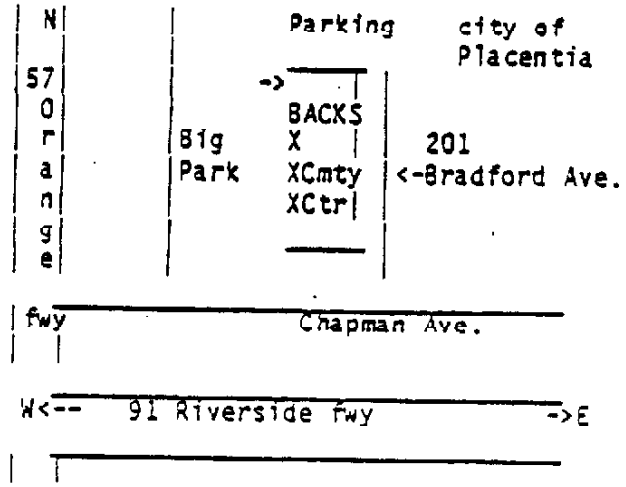
Many product catalogs and flyers will be on-hand and plenty of MICROpendiums.

Win free products donated by the participating commercial enterprizes with the many raffles to be held. And perhaps win the door prize donated by the Imperial Gallery (art).

Sponsored by these groups and their coordinators: Ed Butcher- Riverside group, Gene Bohot- Pomona group, Rodger Merritt and Ken Hamai- Brea group, Jim Swedlow- Orange group, Phil Barnes- El Toro group, Chick DeMarti- Los Angeles group, Tim Remley- Club 99 (Covina), Lee Pierce- San Diego group, Richard Parson- TICO (Oxnard), and Bill Harms, 6527 Hayes Court, Chino, CA 91710, 714-628-1334 eves.

There will be at least 5 presentations (demo.s) by local luminaries of the TI 99-4A world of topics and products of interest.

Come to the FALL-4-A-SHARE-FAIR to get your FAIR SHARE: of bargins, fun, prizes, knowledge! Get that "backup" hardware you always needed.



The BACKS Community Building lower level, room 7 at 201 North Bradford Ave, Placentia, CA 92670. It's at the south east corner of a great park with plenty of grass, shade, barbeque pits, and play equipment. It is on the North side of Chapman Avenue which is just north of the Riverside (91) freeway, and just East of the Orange (57) freway. It's close to the Brea Mall and other stores.